UNIVERSITY OF SASKATCHEWAN

Department of Computer Science

CMPT 215.3 MIDTERM EXAMINATION

March 3rd, 2006

Total Marks: 50

CLOSED BOOK and CLOSED NOTES
NO CALCULATOR

Time: 50 minutes

Instructions

Read each question carefully and write your answer legibly on the examination paper. **No other paper will be accepted**. You may use the backs of pages for rough work but all final answers must be in the spaces provided. The marks for each question are as indicated. Allocate your time accordingly.

Ensure that your name AND student number are clearly written on the examination paper and that your name is on every page.

Note: a reference table of MIPS instructions is provided at the end of the examination paper.

Question	Marks
1 (6 marks)	
2 (12 marks)	
3 (16 marks)	
4 (16 marks)	
Total	

ANSWER KEY

1. **General** (*6 marks in total – 1 mark for each part*) Give the technical term that best fits each of the following descriptions or definitions.

   (a) The laws of this type of algebra are used to simplify logic equations.

   Boolean

   (b) The data structure that is used in the standard MIPS parameter passing convention to pass all but the first four procedure arguments.

   stack

   (c) A digital logic circuit that has three inputs and one output. The output is identical in value to the *first* of the inputs if the *third* input is 0, and is identical in value to the *second* of the inputs if the *third* input is 1.

   multiplexor

   (d) The execution time of each machine language instruction must consist of an integral number of these.

   clock cycles

   (e) A type of adder with which addition of *n* bit numbers takes *O*(*n*) time.

   ripple-carry

   (f) A program chosen to serve as the basis of performance comparison between computer systems.

   benchmark

2. **Computer Performance** (*12 marks in total*)

   (a) (*4 marks*) Suppose that the floating point component of a processor is upgraded so that each floating point instruction takes only 50% as much time to execute as previously. (The execution times of all other types of instructions remain unchanged.) If the upgrade decreases the execution time of a particular program by 20%, what percentage of the original execution time must have been spent on executing floating point instructions?

   Let f be the fraction of the original execution time spent on floating point instructions.

   From Amdahl's Law:

$$\frac{\text{new execution time}}{\text{old execution time}} = \frac{f}{2} + (1 - f)$$

   so 0.8 = f/2 + (1-f), giving f = 0.4, and an answer of 40%.

(b) (*4 marks*) A processor with a clock period length of 0.5 nanoseconds has a CPI of 4 for a particular program. What is the MIPS rating for this program? (Recall that a nanosecond is 1.0 x 10^{-9} seconds.)

$$\text{MIPS} = \frac{\text{clock rate in MHz}}{\text{CPI}} = \frac{\text{clock rate in GHz x 1000}}{\text{CPI}} = 2000 / 4 = 500$$

(c) (*4 marks*) Consider a system with two types of instructions: type *A* instructions each require 5 clock cycles, while type *B* instructions each require 2 clock cycles. Suppose that it is possible to reduce the number of clock cycles required for each class *A* instruction to 4, but at the cost of an increase in the clock period length. If 25% of the instructions in a particular program are of type *A*, and 75% are of type *B*, what would be the maximum factor by which the clock period length could increase, without this modification increasing the program's execution time?

$$\text{maximum factor} = \frac{\text{old CPI}}{\text{new CPI}} = \frac{5 \text{ x } (1/4) + 2 \text{ x } (3/4)}{4 \text{ x } (1/4) + 2 \text{ x } (3/4)} = 1.1$$

3. **Arithmetic** (*16 marks in total*)

(a) (*2 marks*) If Booth's algorithm is used to multiply the 6-bit signed binary number 101010 (multiplicand) by the 6-bit signed binary number 111101 (multiplier), how many times will the multiplicand be added or subtracted?

3 (2 subtractions and 1 addition)

(b) (*6 marks*) Perform the following conversions:

(i) -7_{10} to 5-bit 2's complement. 11001

(ii) F1F4_{16} to binary. 1111 0001 1111 0100

(iii) The number 10001, in 5-bit biased notation with bias of 15, to decimal. 17-15 = 2

Student Number:________________________________

(c) (*4 marks*) Consider the problem of implementing signed overflow detection for subtraction operations. Specifically, we wish to have a 1-bit output that equals 1 if signed overflow occurs on a subtraction operation *a* - *b*, and 0 otherwise. Give a truth table for such a 1-bit output, as a logic function of the high order bits of the operands (a_{31} and b_{31}) and the high order bit of the result ($result_{31}$). Then, using your truth table, derive a logic equation in sum-of-products form.

Note that for addition, have signed overflow if and only if adding two positive numbers gives a negative number, or adding two negative numbers gives a positive number.

Similarly, for subtraction, have signed overflow if and only if subtracting a negative number from a positive number gives a negative number, or subtracting a positive number from a negative number gives a positive number.

a_{31}	b_{31}	$result_{31}$	signed overflow
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

(not a_{31}) b_{31} $result_{31}$ + a_{31} (not b_{31}) (not $result_{31}$)

(d) (*4 marks*) Recall that in the IEEE 754 floating point standard, single precision floating point numbers have a 1-bit sign field, followed by an 8-bit exponent field (in biased notation with a bias of 127), followed by a 23-bit significand field. Give the representation of -13.625_{10}.

First, **separately** convert the integer and fractional parts to base two, which yields the result -1101.101. Then, normalize, yielding 1.101101 x 2^3. 3+127 = 130, which in binary is 10000010. This yields:

1 10000010 10110100 … 0

4. **Machine and Assembly Language** (*16 marks in total*)

(a) (*2 marks*) What is accomplished during each pass of a two-pass assembler?

first pass builds the symbol table, second pass creates the machine language

(b) (*6 marks*) This question concerns the MIPS instructions *beq* and *j*.

(i) Which machine language instruction format does each of these instructions use?

beq: I format; *j*: J format

(ii) In the machine language for *beq*, how exactly is the location to which control should be transferred, in the case that the equality test succeeds, specified?

specified by giving the distance in words from the instruction following the branch

(iii) For instructions of one of these types, special action by the linker is needed, while for instructions of the other type, no such special action is needed. Explain.

When multiple object files are being combined and the linker must shift an object file in the address space, the address field of each *j* machine language instruction must be corrected by adding in the object file's *relocation constant*. For *beq* instructions, no such correction is necessary, since an offset rather than an absolute address is used.

(c) (*8 marks*) Consider a singly-linked list of nodes, where each node consists of two consecutive words containing: 1) an integer value (in the first word), and 2) the memory address of (the first word of) the next node in the list, in the second word. A memory address of zero is used to indicate the end of the list. Write a MIPS procedure *sum* that returns the sum of the integer values of the nodes in the list, when called with the memory address of the first node in the list as a parameter (you may assume that the list has at least one node). You must use the standard procedure call conventions discussed in class. You do NOT need to write a main program.

```
sum:         move $v0, $zero
sum_loop:    lw $t0, 0($a0)
             add $v0, $v0, $t0
             lw $a0, 4($a0)
             bne $a0, $zero, sum_loop
             jr $ra
```

The End